**Coretime Implementation Simulation**

**Run Trappist Zombienet**

1. Clone the trappist repository and compile.

git clonit clone git@github.com:paritytech/trappist.git

cd trappist

cargo build --release

1. Download Binaries

cd zombienet

Download polkadot, polkadot-parachain, polkadot-execute-worker, polkadot-prepare-worker, zombienet-linux-x64

Put the dowloaded binaries in this directory.

1. Download Coretime Rococo Toml

https://gist.github.com/vklachkov/6e2c68a15824d62ce9150f07cfecbc8a

1. Run the Zombienet with coretime\_rococo.toml

./zombienet-linux-x64 –p native spawn ./coretime\_rococo.toml

1. Open the Relay, AssetHub and Trappist in PolkadotJS

**Setup the Relay**

1. On the **Relay** we will do a sudo batch call as follows:

**Pallet:** configuration

**Extrinsic:** setCoretimeCores

**Params:**

**new:** 1 (1 Core to be assigned to the Coretime Chain)

**Pallet:** coretime

**Extrinsic:** assignCore

**Params:**

**core:** 0 (1 Core to be assigned to the Coretime Chain)

**begin:** 50 (a block starting point. Of course needs to be higher if you call this at a higher block number or lower if you need to wait less)

assignment

* Task: 1005 (ParaId of the Coretime chain)
* PartsOf57600: 57600 (therefore, all of it)

**endHint:** None

**Setup Coretime Chain**

1. we move to the **Coretime** Chain where we are going to build a sudo batch call that will execute the following steps:

Configure the broker pallet.

Request a core for the Coretime Chain.

Set a lease for the Coretime Chain with a really long duration so it runs during our whole test.

Start the broker pallet by calling startSales

The call will be composed as follows:

**Pallet:** broker

**Extrinsic:** configure

Params:

**advanceNotice:** 1 (1 Relay Chain block at 6 seg )

**interludeLength:** 2 (1 block at 12 seg)

**leadinLength:** 1 (1 block at 12 seg)

**regionLength:** 2 (2 timeslice or 20 blocks, 1 TIMESLICE = 10 BLOCKS in ROC w/ fast-runtime)

**idealBulkProportion:** 500000000 ( 50% )

**limitCoresOffered:** Null

**renewalBump:** 3500000 (0.00035%)

**contributionTimeout:** 1

1. We will require 2 cores, one for Coretime Chain and one for Trappist.

**Pallet:** broker

**Extrinsic:** requestCoreCount

**Params:**

**coreCount:** 2

1. We set the Coretime Chain itself into a long-lasting lease so it holds for the duration of the test.

**Pallet:** broker

**Extrinsic:** setLease

**Params:**

**task:** 1005

**until:** 1000

1. We start the sales with one core offered for Trappist to be able to buy.

**Pallet:** broker

**Extrinsic:** startSales

**Params:**

InitialPrice 1000000 (just an arbitrary value)

CoreCount 1

**Buy Bulk Coretime**

1. Since we are working with tight timeframes, we will batch the core purchasing core and the assignment like this:

**Pallet:** broker

**Extrinsic:** purchase

**Params:**

priceLimit: 10000000 (any big number, bigger than the price)

**Pallet:** broker

**Extrinsic:** assign

**Params:**

regionId

**begin:** X

**core:** 1

**mask:** 0xffffffffffffffffffff

**task:** 1000

**finality:** Final

1. We can check the coretime being assigned in real time by checking the chain state of broker under workload for Core 1
2. If we also want to check this assignment from Relay side, we can check coreDescriptors of coretimeAssignmentProvide for Core 1.

**Renew a Core**

1. **Pallet**: broker

**Extrinsic:** renew

**Params:**

**core:** 1 (Core number to renew)

**Renew a core via XCM**

1. **Pallet:** xcmPallet

**Extrinsic:** limitedTeleportAsset

**Params:**

**dest:** V3 (Coretime Chain from Relay perspective)

{0,X1(Parachain(1005))}

**beneficiary:** V3 (Trappist’s Sovereign Account in Coretime Chain from Coretime Chain perspective)

{0,X1(AccountId32( "TRAPPIST SA ADDRESS" ))}

**assets:** V3 (ROC from Relay perspective)

**id:** Concrete {0, Here}

**fun:** 1000000000000

**feeAssetItem:** 0

**weightLimit:** Unlimited

1. Once the core is eligible for renewal, here is the extrinsic that must be called with sudofrom Trappist:

**Pallet:** polkadotXcm

**Extrinsic:** send

**Params:**

**dest:** V3 (Coretime Chain from Trappist perspective)

{1 , X1(Parachain(1005))}

**message:** V3 (XCM logic required for the Transact instruction)

WithdrawAsset (Withdraw asset on destination to pay execution)

id : {1 , Here } (ROC on coretime)

**fun:** 100000000000

BuyExecution (Buy execution on destination)

id : {1 , Here } (ROC on Coretime)

**fun:** 100000000000

Transact (Execute an extrinsic on destination)

**originKind:** SovereignAccount (We execute it with Trappist Sovereign Account as origin)

requireWeightAtMost : (8000000000, 65536) (Arbitrary enough weight limit)

**call:** 0x32060100 (This is the renew call from broker pallet on Coretime Chain itself which in this case is for Core 1 since is the one assigned to Trappist )

RefundSurplus (Refund what is left from execution cost)

DepositAsset (Into Trappist’s SA on Coretime Chain)

filter : (Wild, All)

**beneficiary:** {0,X1(AccountId32( "TRAPPIST SA ADDRESS" ))}